Unification-Based Grammar Engineering

Dan Flickinger
Stanford University & Redbird Advanced Learning
danf@stanford.edu

Stephan Oepen
Oslo University
oe@ifi.uio.no

ESSLLI 2016; August 15–19, 2016

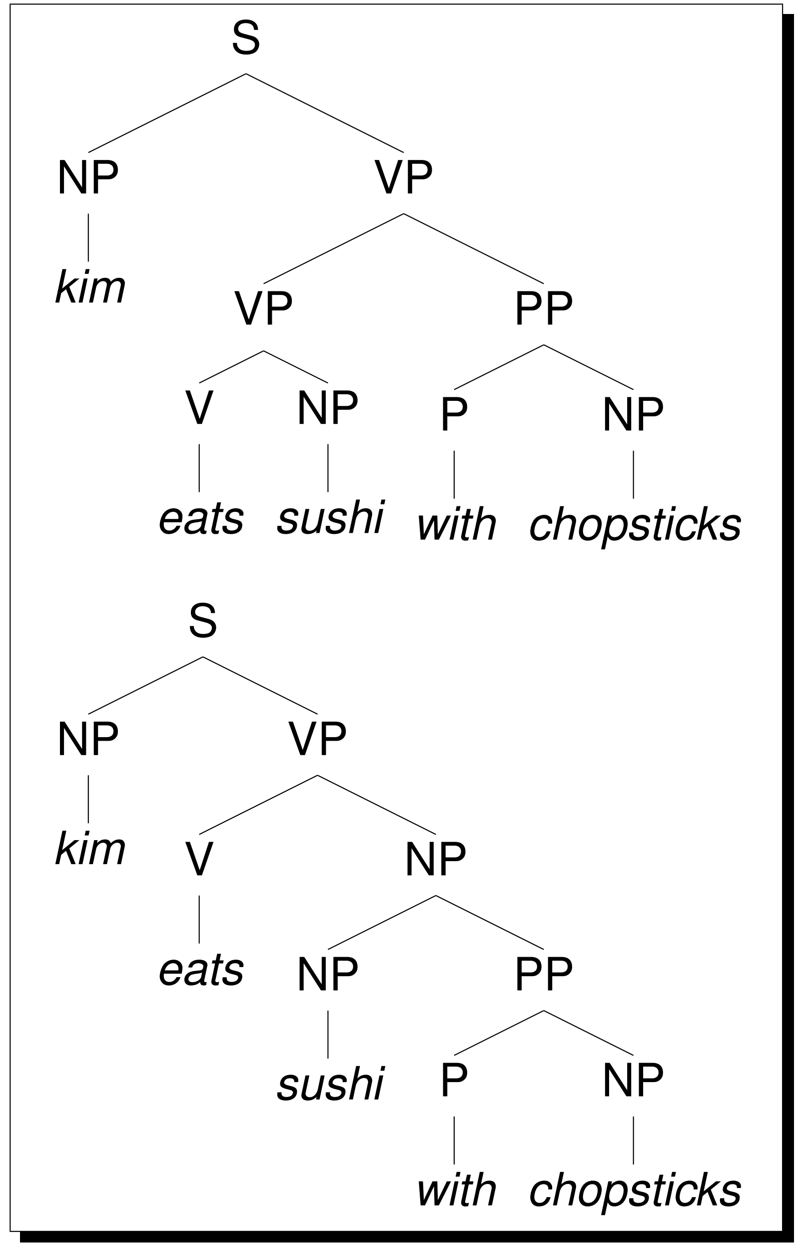
Recognizing the Language of a Grammar $\langle C, \Sigma, P, S \rangle$

P :

S $\rightarrow$ NP VP
VP $\rightarrow$ V NP
VP $\rightarrow$ VP PP
NP $\rightarrow$ NP PP
PP $\rightarrow$ P NP
NP $\rightarrow$ kim | sushi | chopsticks
V $\rightarrow$ snores | eats
P $\rightarrow$ with

All Complete Derivations

- are rooted in the start symbol S;
- label internal nodes with categories $\in C$, leafs with words $\in \Sigma$;
- instantiate a grammar rule $\in P$ at each local subtree of depth one.



Limitations of Context-Free Grammar

Agreement and Valency (For Example)

That dog barks.

**That dogs barks.*

**Those dogs barks.*

The dog chased a cat.

**The dog barked a cat.*

**The dog chased.*

**The dog chased a cat my neighbors.*

The cat was chased by a dog.

**The cat was chased of a dog.*

...

Structured Categories in a Unification Grammar

- All (constituent) categories in the grammar are typed feature structures;
- specific TFS configurations may correspond to 'traditional' categories;
- labels like 'S' or 'NP' are mere abbreviations, not elements of the theory.

$$\mathit{word}\begin{bmatrix} \mathrm{HEAD} & \mathit{noun} \\ \mathrm{SPR} & \left\langle \begin{bmatrix} \mathrm{HEAD}\ \mathit{det} \end{bmatrix} \right\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix} \qquad \mathit{phrase}\begin{bmatrix} \mathrm{HEAD} & \mathit{verb} \\ \mathrm{SPR} & \langle\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix} \qquad \mathit{phrase}\begin{bmatrix} \mathrm{HEAD} & \mathit{verb} \\ \mathrm{SPR} & \left\langle \begin{bmatrix} \mathrm{HEAD}\ \mathit{noun} \end{bmatrix} \right\rangle \\ \mathrm{COMPS} & \langle\rangle \end{bmatrix}$$

'N'	'S'	'VP'
'lexical'	'maximal'	'intermediate'

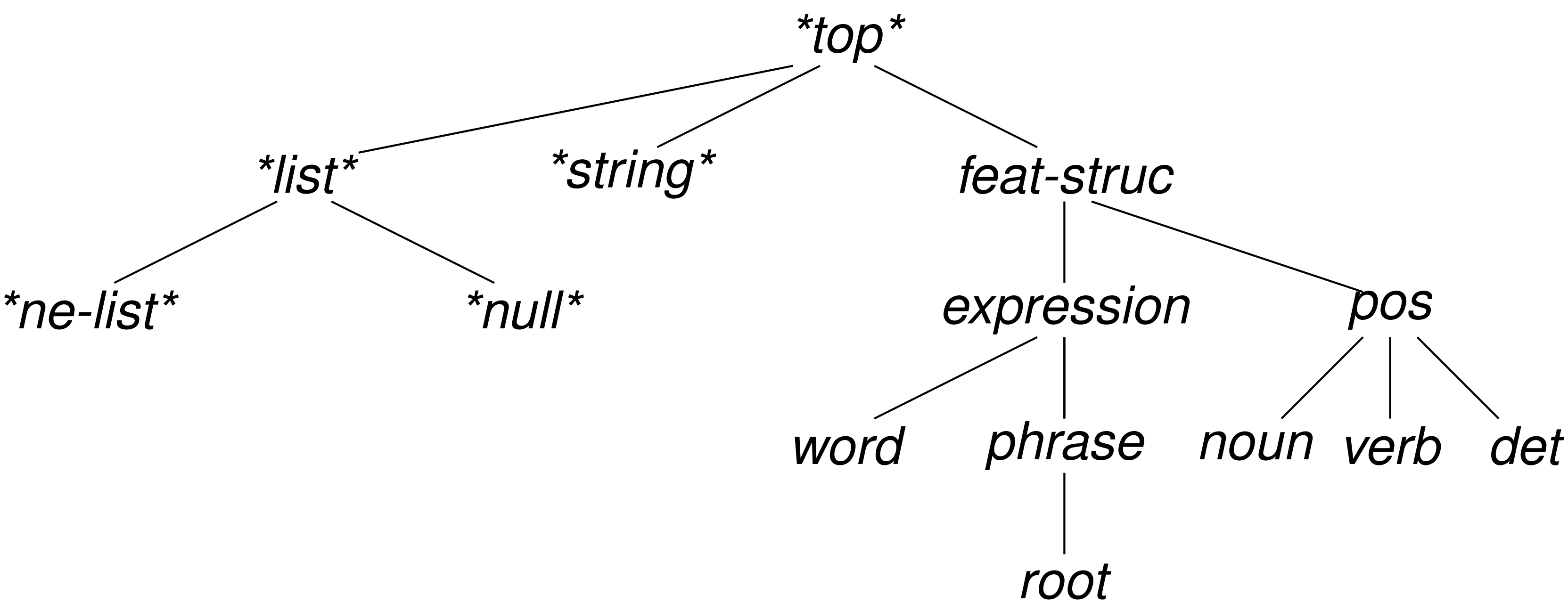
Interaction of Lexicon and Phrase Structure Schemata

$$
\mathit{phrase}\begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{SPR} & \langle\rangle\\ \text{COMPS} & \boxed{3}\end{bmatrix} \longrightarrow \boxed{2}\,\mathit{phrase}\begin{bmatrix}\text{SPR} & \langle\rangle\\ \text{COMPS} & \langle\rangle\end{bmatrix}, \quad \mathit{phrase}\begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{SPR} & \langle\boxed{2}\rangle\\ \text{COMPS} & \boxed{3}\end{bmatrix}
$$

$$
\mathit{phrase}\begin{bmatrix}\text{ORTH} & \textit{“the dog”}\\ \text{HEAD} & \mathit{noun}\begin{bmatrix}\text{AGR} & \mathit{3sg}\end{bmatrix}\\ \text{SPR} & \langle\rangle\\ \text{COMPS} & \langle\rangle\end{bmatrix}
\qquad
\mathit{phrase}\begin{bmatrix}\text{ORTH} & \textit{“barks”}\\ \text{HEAD} & \mathit{verb}\begin{bmatrix}\text{AGR} & \boxed{1}\,\mathit{3sg}\end{bmatrix}\\ \text{SPR} & \left\langle\begin{bmatrix}\text{HEAD} & \mathit{noun}\begin{bmatrix}\text{AGR} & \boxed{1}\end{bmatrix}\\ \text{SPR} & \langle\rangle\\ \text{COMPS} & \langle\rangle\end{bmatrix}\right\rangle\\ \text{COMPS} & \langle\rangle\end{bmatrix}
$$

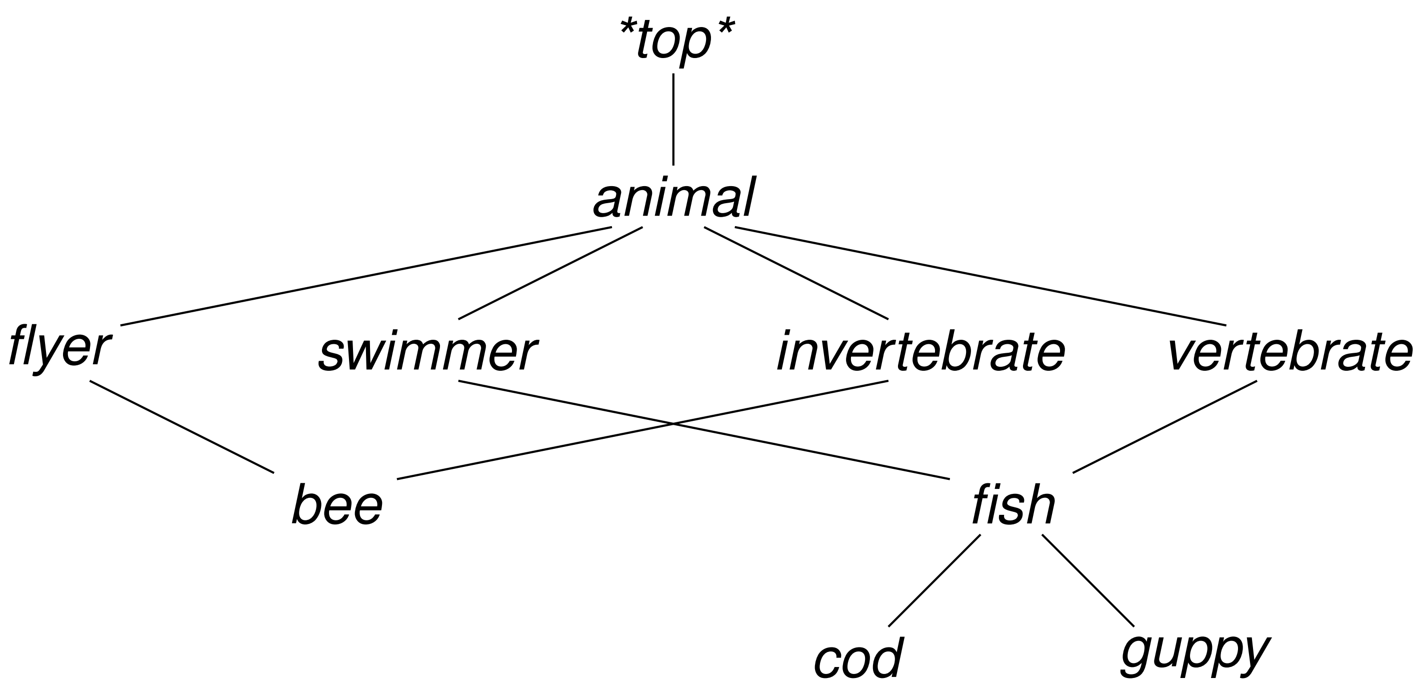
The Type Hierarchy: Fundamentals

- Types ‘represent’ groups of entities with similar properties (‘classes’);
- types ordered by specificity: subtypes inherit properties of (all) parents;
- type hierarchy determines which types are compatible (and which not).



Multiple Inheritance

- *flyer* and *swimmer* no common descendants: they are incompatible;
- *flyer* and *bee* stand in hierarchical relationship: they unify to subtype;
- *flyer* and *invertebrate* have a unique greatest common descendant.



Typed Feature Structure Subsumption

- Typed feature structures can be partially ordered by information content;
- a more general structure is said to *subsume* a more specific one;
- $\textit{*top*}\begin{bmatrix} \end{bmatrix}$ is the most general feature structure (while $\bot$ is inconsistent);
- $\sqsubseteq$ ('square subset or equal') conventionally used to depict subsumption.

Feature structure F subsumes feature structure G ($F \sqsubseteq G$) iff: (1) if path p is defined in F then p is also defined in G and the type of the value of p in F is a supertype or equal to the type of the value of p in G, and (2) all paths that are reentrant in F are also reentrant in G.

Feature Structure Subsumption: Examples

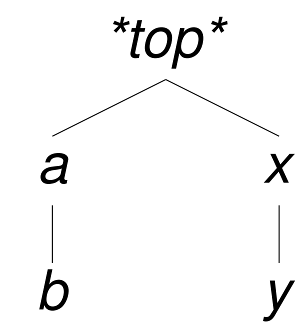
TFS$_1$: $\left[\begin{array}{l} \text{FOO } x \\ \text{BAR } x \end{array}\right]_a$

TFS$_2$: $\left[\begin{array}{l} \text{FOO } x \\ \text{BAR } y \end{array}\right]_a$

TFS$_3$: $\left[\begin{array}{l} \text{FOO } y \\ \text{BAR } x \\ \text{BAZ } x \end{array}\right]_b$

TFS$_4$: $\left[\begin{array}{l} \text{FOO } \boxed{1}\, x \\ \text{BAR } \boxed{1} \end{array}\right]_a$

Signature



Feature structure F subsumes feature structure G ($F \sqsubseteq G$) iff: (1) if path p is defined in F then p is also defined in G and the type of the value of p in F is a supertype or equal to the type of the value of p in G, and (2) all paths that are reentrant in F are also reentrant in G.

Typed Feature Structure Unification

- Decide whether two typed feature structures are mutually compatible;
- determine combination of two TFSs to give the most general feature structure which retains all information which they individually contain;
- if there is no such feature structure, unification fails (depicted as $\perp$);
- unification *monotonically* combines information from both 'input' TFSs;
- *relation to subsumption* the unification of two structures F and G is the most general TFS which is subsumed by both F and G (if it exists).
- $\sqcap$ ('square set intersection') conventionally used to depict unification.

Typed Feature Structure Unification: Examples

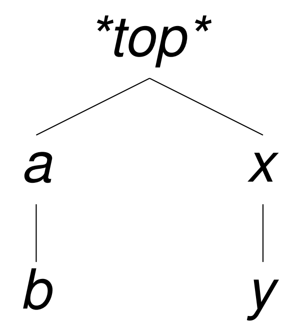
TFS_1: $_{a}\begin{bmatrix}\text{FOO} & x \\ \text{BAR} & x\end{bmatrix}$

TFS_2: $_{a}\begin{bmatrix}\text{FOO} & x \\ \text{BAR} & y\end{bmatrix}$

TFS_3: $_{b}\begin{bmatrix}\text{FOO} & y \\ \text{BAR} & x \\ \text{BAZ} & x\end{bmatrix}$

TFS_4: $_{a}\begin{bmatrix}\text{FOO} & \boxed{1}\, x \\ \text{BAR} & \boxed{1}\end{bmatrix}$

Signature



$\text{TFS}_1 \sqcap \text{TFS}_2 \equiv \text{TFS}_2$ $\quad$ $\text{TFS}_1 \sqcap \text{TFS}_3 \equiv \text{TFS}_3$ $\quad$ $\text{TFS}_3 \sqcap \text{TFS}_4 \equiv {}_{b}\begin{bmatrix}\text{FOO} & \boxed{1}\, y \\ \text{BAR} & \boxed{1} \\ \text{BAZ} & x\end{bmatrix}$

Notational Conventions

- lists not available as built-in data type; abbreviatory notation in TDL:

```
< a, b > ≡ [ FIRST a, REST [ FIRST b, REST *null* ] ]
```

- underspecified (variable-length) list:

```
< a ... > ≡ [ FIRST a, REST *list* ]
```

- difference (open-ended) lists; allow concatenation by unification:

```
<! a !> ≡ [ LIST [ FIRST a, REST #tail ], LAST #tail ]
```

- built-in and ‘non-linguistic’ types pre- and suffixed by asterisk (**top**);
- strings (e.g. *“chased”*) need no declaration; always subtypes of **string**;
- strings cannot have subtypes and are (thus) mutually incompatible.